Finding Isolated Cliques by Queries – An Approach to Fault Diagnosis with Many Faults

William Gasarch[1] and Frank Stephan[2]

[1] University of Maryland, Department of Computer Science and Institute of Advanced Studies, A.V. Williams Building, College Park, MD 20742, USA
gasarch@cs.umd.edu

[2] National University of Singapore, Departments of Computer Science and Mathematics, 3 Science Drive 2, Singapore 117543, Republic of Singapore
fstephan@comp.nus.edu.sg

Abstract. A well-studied problem in fault diagnosis is to identify the set of all good processors in a given set $\{p_1, p_2, \ldots, p_n\}$ of processors via asking some processors p_i to test whether processor p_j is good or faulty. Mathematically, the set C of the indices of good processors forms an isolated clique in the graph with the edges $E = \{(i, j)$: if you ask p_i to test p_j then p_i states that "p_j is good"$\}$; where C is an isolated clique iff it holds for every $i \in C$ and $j \neq i$ that $(i, j) \in E$ iff $j \in C$.
In the present work, the classical setting of fault diagnosis is modified by no longer requiring that C contains at least $\frac{n+1}{2}$ of the n nodes of the graph. Instead, one is given a lower bound a on the size of C and the number n of nodes and one has to find a list of up to n/a candidates containing all isolated cliques of size a or more where the number of queries whether a given edge is in E is as small as possible.
It is shown that the number of queries necessary differs at most by n for the case of directed and undirected graphs. Furthermore, for directed graphs the lower bound $n^2/(2a-2) - 3n$ and the upper bound $2n^2/a$ are established. For some constant values of a, better bounds are given. In the case of parallel queries, the number of rounds is at least $n/(a-1) - 6$ and at most $O(\log(a)n/a)$.

Keywords: Isolated Cliques, Query-Complexity, Fault Diagnosis

1 The Starting Point: Diagnosing Faulty Processors

Fault diagnosis can be studied on an abstract level: Given a network of n processors $p_1, p_2, \ldots, p_n$, determine which of them are working correctly, that is, find the good processors represented by the set

$$C = \{i : \text{ the } i\text{-th processor } p_i \text{ is not faulty}\}.$$

The task has to be solved by asking processors p_i to test whether p_j is correct. There are four possibilities what can happen if p_i is requested to test p_j. If p_i

and p_j are both good, then p_i verifies this by an adequate test and outputs 1 meaning "good". If p_i is good and p_j is faulty, then p_j will not pass the test given by p_i and p_i will output 0 meaning "faulty". If p_i is faulty, then p_i might test p_j or not and might output whatever it wants, but without loss of generality one can identify illegal output with 0 and restrict the range of these outputs to $\{0, 1\}$ as well.

Note that the faulty processors may conspire to make testing as difficult as possible. Hence lower bounds are proven by describing a strategy for the faulty processors. In particular, every faulty processor would consider itself to be good and so self-tests will always result in the output 1. Therefore, self-tests are not used in this framework.

This general scenario of asymmetric fault-diagnosis has been studied by Malek [9] while Chwa and Hakimi [7] imposed the restriction that p_i and p_j judge each other in the same way: if p_i says that p_j is faulty, so does p_j. The classical approach is where the majority of the processors in not faulty. Then one can determine the set of good processors with a short number of queries. Blecher [4] determined this answer exactly and it does not only work in the symmetric model of Chwa and Hakimi [7] but also in the asymmetric of Malek [9]. In the following, b is an upper bound on the number of faulty processors among n processors $p_1, p_2, \ldots, p_n$.

Classical Fault Diagnosis Problem. Given n processors $p_1, p_2, \ldots, p_n$ and an upper bound b on the number of faulty processors where $b < \frac{n}{2}$, what is the number m in dependence on n and b such that an algorithm can find with asking for at most m tests the set $C = \{i : p_i \text{ is good}\}$ of (indices of) good processors.

For the classical fault diagnosis problem, the number of queries turned out to be the same in the asymmetric and symmetric model; Blecher [4] determined the number of queries exactly.

Fact 1.1 [4]**.** *If at most b out of n processors are faulty and if $b < \frac{1}{2} \cdot n$ then one can determine the set C of (indices of) good processors with at most $n + b - 1$ many queries (= tests). This bound $n + b - 1$ is tight and cannot be improved.*

This result was extended by considering the case where the tests are performed in parallel: If i, j, i', j' are four different numbers in $\{1, 2, \ldots, n\}$ then the processor p_i can test p_j and the processor $p_{i'}$ can test $p_{j'}$ without any interference. So it is natural to ask, how many rounds are needed for fault diagnosis and the answer was the following.

Fact 1.2 [2, 3, 8]**.** *If the absolute majority of processors is good, one can determine the set C of (indices of) good processors with at least 5 and at most 10 rounds of parallel tests such that every processor is involved in at most one test per round either as a testing or as a tested processor.*

The starting point of the present work is the question what happens if the condition that the majority of the processors is good is dropped. Of course, one cannot determine the set C of good processors itself as the faulty processors can fool out the tester by just claiming that every good processor would be faulty and every faulty one would be good. So one would consider the following modified fault diagnosis problem where again b is an upper bound on the number of faulty processors.

Modified Fault Diagnosis Problem. What are the numbers m, d in dependence on an upper bound b on the number of faulty processors and the number n of processors $p_1, p_2, \ldots, p_n$ such that an algorithm can find with asking for at most m tests a list of d candidate-sets containing also the set $C = \{i : p_i$ is good$\}$ of good processors and that there is no such algorithm with $d-1$ in place of d.

The modified fault diagnosis problem turns out to be equivalent to the graph theoretic problem of finding isolated cliques.

Definition 1.3. Given a graph $G = (\{1, 2, \ldots, n\}, E)$, a set C of nodes is called an isolated clique iff for every node $i \in C$ and every node $j \in \{1, 2, \ldots, n\} - \{i\}$ it holds that $(i, j) \in E$ iff $j \in C$. Now the following numbers to measure the complexity of finding a list of all isolated cliques are defined.

1. $D(a, n)$ is the minimal number of queries needed to determine a list of up to $\lfloor \frac{n}{a} \rfloor$ many members containing all the isolated cliques of size at least a where G ranges over all directed graphs of at least n nodes and having at least one isolated clique. (The D stands for Directed.)
2. $U(a, n)$ is the minimal number of queries needed to determine a list of up to $\lfloor \frac{n}{a} \rfloor$ many members containing all the isolated cliques of size at least a where G ranges over all undirected graphs of at least n nodes and having at least one isolated clique. (The U stands for Undirected.)
3. $PU(a, n)$ is the minimal number of rounds of queries needed to determine a list of up to $\lfloor \frac{n}{a} \rfloor$ many members containing all the isolated cliques of size at least a where G ranges over all undirected graphs of at least n nodes and having at least one isolated clique. (The PU stands for Parallel Undirected.)

Then $D(a, n)$ is the directed graph approach of Malek [9] and $U(a, n)$ is the undirected graph approach of Chwa and Hakimi [7]. $PU(a, n)$ is the corresponding model for parallel queries. Since there are no nontrivial results for parallel directed queries which cannot derived from the other cases easily, this notion is left out in this paper.

Note that the definitions of the above notions are robust in the sense that one could also include the case where G does not have any isolated clique. The reason is that one can just stop the algorithm without any further output before asking more queries than permitted or outputting more candidates than permitted as one knows that in this case there is no isolated clique. Furthermore, one can list a candidate but cannot establish that this candidate is indeed an isolated clique as this additional procedure would require at least $\frac{a(a-1)}{2}$ many queries. Therefore also illegal candidates are permitted in the list output as long as these candidates do not break the given bound on the cardinality of the list. The next proposition gives just a formal verification for the approach to use graph-theoretic tools for fault diagnosis [10, 11].

Proposition 1.4. *Given a Modified Fault Diagnosis Problem where a is the minimum number of good processors and n is the number of all processors, the maximal number m of required tests is exactly $D(a,n)$ and the size d of the output list is $\lfloor \frac{n}{a} \rfloor$.*

Proof. Given an algorithm for the Modified Fault Diagnosis Problem with parameters n and a and always outputting a list of the minimum number of hypotheses necessary, let $C \subseteq \{1,2,\ldots,n\}$ be that set of good processors containing at least a nodes on which the algorithm has it worst case performance and let R be a run of the algorithm leeding to these m queries. In this run R it is assumed that no query is done twice – a processor answering the same query twice with a different output would give away that it is faulty while repeating the last answer would not reveal any new knowledge to the algorithm. Thus one can assume that every query in the protocol R is asked exactly once. The algorithm needs m queries for the run R and the list of sets output by the algorithm has at most $\frac{n}{a}$ many members. Now one puts into E all pairs (i,j) such that either the run R of the algorithm requested p_i to test p_j and received the answer 1 or that i,j are both in C and $i \neq j$. Then C is an isolated clique of the graph $(\{1,2,\ldots,n\},E)$ and the algorithm could be looked upon as an algorithm to find all isolated cliques. As any node in an isolated clique C uniquely determines C, any two isolated cliques are disjoint and so there can be at most $\frac{n}{a}$ of them, thus one can request that the algorithm outputs at most $\frac{n}{a}$ many candidates.

On the other hand, one can for every graph $(\{1,2,\ldots,n\},E)$ run any algorithm for the Modified Fault Diagnosis Problem by supplying a 1 iff the requested test (p_i,p_j) is represented by an edge $(i,j) \in E$ and by replying a 0 otherwise. As any isolated clique of size a can represent the set of good processors, the algorithm has to output a candidate for every isolated clique.

Let $d = \lfloor \frac{n}{a} \rfloor$, that is, d is the maximal integer with $d \cdot a \leq n$. On one hand, the graph where $(i,j) \in E$ iff $i \equiv j$ modulo d partitions the set $\{1,2,\ldots,n\}$ into d cliques of at least a nodes. Each of these isolated cliques can represent the set of good processors. Thus any algorithm needs to output at least d candidates of for the set of good processors, namely these d cliques. On the other hand, if one asks all queries one will recover the graph completely and each candidate for the set of good processors corresponds to an isolated clique of the graph. Since these cliques are disjoint, there are at most d of them having the size a or more. Thus the calculated value d is the optimal possible size of the output list. ∎

Summary of Main Results. In Theorem 2.1 a close connection between the cases for directed and undirected graphs is shown, namely $U(a,n) \leq D(a,n) \leq U(a,n)+n$. Therefore the subsequent results address only the case of undirected graphs.

In Theorems 3.1 and 3.2 it is shown that $\frac{1}{2(a-1)}n^2 - 3n \leq U(a,n) \leq \frac{2}{a}n^2$ for all a,n with $2 \leq a \leq n$. These bounds are quite close, the remaining multiplicative gap is approximately $4\frac{a}{a-1}$.

For constant values of a the bounds on $U(a,n)$ can be tightened. Theorem 4.1 gives for $U(2,n)$ the precise number $\lceil \frac{n(n-2)}{2} \rceil$.

Finally, for the case of parallel queries, the lower bound of over $\frac{n}{a-1} - 6$ many rounds follows directly from Theorem 3.2 while unfortunately no upper bound better than $PU(a,n) \in O(\frac{n \log a}{a})$ was obtained.

2 Isolated Cliques of Undirected Graphs

In the following it is shown that the values of $D(a,n)$ and $U(a,n)$ for directed and undirected graphs are very near to each other. Therefore, further research will then deal with $U(a,n)$ instead of $D(a,n)$.

Theorem 2.1. $U(a,n) \leq D(a,n) \leq U(a,n) + n$.

Proof. The first relation "$\leq$" is straight forward. So only the second one has to be proven. The idea is to simulate a program M for undirected graphs and to supply its queries with answers sufficiently near to the situation in the given directed graph $(D, \{1,2,\ldots,n\})$. In particular, any isolated clique of D of size a or more should also be consistent with all information fed into M. So the simulating algorithm constructs together with the simulation a graph $(\{1,2,\ldots,n\}, E)$ where for an edge is fixed to be in E only iff M requests to know whether this edge is in E or not. Without loss of generality, M never asks the same query twice. Now it is shown how to construct an algorithm N for the directed case which mainly simulates M.

For bookkeeping, N has an equivalence relation $\equiv$ and a set T. One can always unite equivalence classes and put nodes into T. For every set C which is a subset of some equivalence class of $\equiv$ at some time, one knows that either C is a subset of an isolated clique or C is disjoint to all isolated cliques of $(\{1,2,\ldots,n\}, D)$. Furthermore T is also disjoint to all isolated cliques of $(\{1,2,\ldots,n\}, D)$.

- At the beginning of the simulation of M, N defines that $i \equiv j$ iff $i = j$ and $T = \emptyset$.
- Whenever during the simulation, M queries whether $(i,j) \in E$, then N answers as described below, updates $\equiv$, updates T and then continues the simulation until M terminates. After termination of the simulation, N copycats the output of M without change.

The query whether $(i,j) \in E$ is answered according to the first below case to apply.

1. $i \in T$ or $j \in T$: The answer is "NO" and no updating is done.
2. $i \equiv j$: The answer is "YES" and no updating is done.
3. $(i,j) \notin D$: The answer is "NO" and no updating is done.
4. $(i,j) \in D \wedge (j,i) \notin D$: The answer is "NO" and T is updated by putting all nodes i' with $i' \equiv i$ into T.
5. $(i,j) \in D \wedge (j,i) \in D$: The answer is "YES" and one updates $\equiv$ by uniting the equivalence classes of i, j: after the update $i' \equiv j'$ if before the update either $i' \equiv i \wedge j' \equiv j$ or $i' \equiv j \wedge j' \equiv i$ or $i' \equiv j'$.

Note that a query of M is translated into two queries of N if and only if it is answered by cases 4 and 5. Otherwise it is translated into at most one query. In both cases, the number of equivalence classes outside T goes down by one: in case 4, two equivalence classes are united, in case 5, one is moved into T. Since there are at the beginning n equivalence classes, there can be at most n double queries and $D(a,n) \leq U(a,n) + n$. So it remains to show the correctness of the algorithm.

Let E be a set of undirected edges such that $(i,j) \in E$ iff either i,j belong to the same isolated clique of $(\{1,2,\ldots,n\},D)$ or (i,j) was queried by the simulated machine M in the above algorithm and received the answer "YES". It is shown that the following properties hold for every isolated clique C of $(\{1,2,\ldots,n\},D)$:

- If $i' \in C$ and $i' \equiv j'$ at some stage then $j' \in C$: This can be seen by induction over the running time of the algorithm. So assume that it holds up to the time when i', j' become equivalent. There are i, j such that $i' \equiv i$ and $j' \equiv j$ before the update and $(i,j),(j,i) \in D$. By induction hypothesis, $i \in C$. Since C is a isolated clique, also $j \in C$. Again by induction hypothesis, $j' \in C$.
- If $i' \in T$ at some stage than $i' \notin C$. At the stage where i' goes into T there are i,j such that $i \equiv i'$ and $(i,j) \in D \wedge (j,i) \notin D$. So if an isolated clique contains i it also contains j; but then there are members of it which are not connected in the directed graph, a contradiction. Hence $i \notin C$ and by the previous item, also $i' \notin C$.
- C is an isolated clique of $(\{1,2,\ldots,n\},E)$: Let $i \in C$. One has to show that $j \in C \Leftrightarrow (i,j) \in E$. The direction ($\Rightarrow$) holds already by the definition of E. The direction ($\Leftarrow$) follows from the fact that a "YES" is given according to cases 2 and 5 where in case 2 $i \equiv j$ holds already before giving the answer and in case 5 $i \equiv j$ holds after giving the answer. Since $i \equiv j$ can at any time only hold for $j \in C$, $(i,j) \in E$ only for $j \in C$.

It remains to show that the answers used in the simulation of M are consistent with the graph $(\{1,2,\ldots,n\},E)$. Since M does never query an edge twice, inconsistencies cannot come from supplying different answers to the same query. Without loss of generality, M also never queries first (i,j) and then (j,i). If M receives the answer "YES" then $(i,j) \in E$ by definition. So one has only to check that $(i,j) \notin E$ for the case that M receives the answer "NO". If i,j do not belong to the same isolated clique of D then that answer is correct. If i,j belong to an isolated clique of D then this clique is disjoint to T during the whole time of the algorithm. So the "NO" cannot come from cases 1,3,4 and the used in the simulation of M is the correct answer "YES". Thus while simulating the algorithm M, all answers used are consistent with E and M outputs a list containing all isolated cliques of $(\{1,2,\ldots,n\},E)$ which have at least size a. Every clique C of $(\{1,2,\ldots,n\},D)$ having at least a nodes is in this list and thus N is correct. ∎

Note that the given algorithm runs in time polynomial in n whenever the simulated algorithm M has the same property. The above result can be used to get a non-trivial bound for the number of inspected edges in the case of a directed graph.

Proposition 2.2. $U(1,n) = \frac{1}{2}n(n-1)$. $D(1,n) \leq \frac{1}{2}n(n+1)$.

Proof. Clearly if one knows all edges of a graph, one can compute all isolated cliques of the graph. On the other hand, if an algorithm to find the isolated cliques of the graph $(\{1,2,\ldots,n\},D)$ always receives the negative answer $(i,j) \notin D$ to its queries, then it has at the end to conjecture every singleton $\{i\}$ as a possible isolated clique and also every pair $\{i,j\}$ where it has not asked whether $(i,j) \in D$ or not. The resulting number of cliques is n plus the number of non-asked edges, but as only n hypotheses are permitted, each edge must have been queried. The bound for $D(1,n)$ is then a direct corollary from Theorem 2.1. ∎

3 General Bounds for U(a,n)

This section gives general lower and upper bounds for $U(a,n)$ which match up to a multiplicative constant. In the next section, the upper bound is improved for fixed values of a and exact bounds are given for $a = 2$.

Theorem 3.1. $U(a,n) \leq \frac{3}{2 \cdot a}n^2$.

Proof. Let $(\{1,2,\ldots,n\},E)$ be the given graph and let $d = \lfloor \frac{n}{a} \rfloor$. The idea of the algorithm is the following: One computes a series of sets $D_0, D_1, \ldots, D_n$ such that each set D_j represents members of different possible isolated cliques. In parallel, one builds for each i a set F_i which contains candidates of nodes in the same isolated clique as i. For each new node j one either finds some $i \in D_{j-1}$ which is connected to j. Then one puts j into F_i and defines $D_j = D_{j-1}$. Or one has a new candidate for an isolated clique and defines $D_j = D_{j-1} \cup \{j\}$ unless some exception handling is necessary because D_j would be too big. The exception handling avoids that D_j has more than d elements. It is done such that G_j is the accumulation of all nodes covered by the exception handling and it takes always either 0 or $d+1$ new nodes into G_j; in the second case each isolated clique contributes at most one node to $G_j - G_{j-1}$. Based on this fact one can use cardinality arguments to show that G_j will not contain all elements of any isolated clique of a or more elements. All variables are directly known completely in the stage where they are constructed, only j is put into some F_i in round j of step (2) so that the F_i are constructed during the whole step (2). Thus the variable $u_{i,j}$ considers only $\ell \leq j$ when the value depends on the question which of the ℓ are in F_i. Now the algorithm is presented formally.

(1) Let $D_0 = \emptyset$ and $G_0 = \emptyset$.
(2) For $j = 1,2,\ldots,n$ do the following substeps:
   - Ask for all $i \in D_{j-1}$ whether $(i,j) \in E$; this is void if $D_{j-1} = \emptyset$.
   - Let $j \in F_i$ for the minimal $i \in D_{j-1} \cup \{j\}$ such that either $(i,j) \in E$ or $j = i$; let $j \notin F_i$ for all other i.
   - If $j \notin F_j$ then let $U_j = \emptyset$, $D_j = D_{j-1}$, $G_j = G_{j-1}$.
   - If $j \in F_j$ and $|D_{j-1}| < d$ then let $D_j = D_{j-1} \cup \{j\}$, $G_j = G_{j-1}$ and $U_j = \emptyset$.

- If $j \in F_j$ and $|D_{j-1}| = d$ then let $u_{i,j} = \{\max\{\ell \in F_i : \ell \leq j \wedge \ell \notin G_{j-1}\}$ for all $i \in D_{j-1} \cup \{j\}\}$, $G_j = G_{j-1} \cup \{u_{i,j} : i \in D_{j-1} \cup \{j\}\}$ and $D_j = D_{j-1} - G_j$.

(3) For every $i \in D_n$, let C_i be the union of all F_j where either $j = i$ or $j \in G_n \wedge j \in F_j \wedge j < i \wedge (j,i) \in E$ where C_i is obtained by asking whether $(j,i) \in E$ for those j satisfying $j \in G_n \wedge j \in F_j \wedge j < i$.

First the bound on the number of queries is verified. It is easy to see by induction that $|D_j| \leq d$ for all j. Thus, in the j-th round of step (2), there are at most d queries. This gives $d \cdot n$ queries in total for step (2). In step (3), there are up to d elements i in D_n and for each C_i is computed by making up to $|G_n|$ queries. So the overall number of queries is bounded by $(n + |G_n|) \cdot d$. But this bound can be improved to $(n + \frac{1}{2} \cdot |G_n|) \cdot d$ as shown in the next paragraph. This number is bounded by $\frac{3}{2 \cdot a} n^2$ since $d \leq \frac{n}{a}$ and $|G_n| \leq n$.

The idea is that whenever one puts some elements into G_n, one can argue that based on some conditions, certain queries will not be made later so that one can substract some number from the rough upper bound $(n + |G_n|) \cdot d$. As a consequence one can conclude that the algorithm makes never more than $(n + \frac{1}{2}|G_n|) \cdot d$ queries. If at stage j it holds that $G_j \neq G_{j-1}$ then let e_j be the number of $u_{k,j}$ in $G_j - G_{j-1}$ with $k < u_{k,j}$. When constructing C_i in step (3), no query of the form $(i, u_{k,j})$ will be made since $u_{k,j} \in F_k$ for some k different from $u_{k,j}$. So one saves $d \cdot e_j$ queries which can be subtracted from the bound $|G_n| \cdot d$. Furthermore, $|D_j| = d - e_j$. So one knows that G_k will have at most $d - e_j + k - j$ elements for $k = j, j+1, \ldots, j+e_j-k$. In the case that $G_j \neq G_n$ one can conclude that for the computation of $D_{j+1}, \ldots, D_{d+1-e_j}$ at least $\frac{1}{2}(d - e_j)(d + 1 - e_j)$ are saved compared to the estimated bound since $j + (d + 1 - e_j) \leq n$. In the case that $G_j = G_n$ one can make use of the fact that when computing $D_1, \ldots, D_d$, actually at least $\frac{1}{2}d(d+1)$ queries are saved compared to the actual bound estimated for these queries. So one can conclude that for each $j \in \{1, 2, \ldots, n\}$ with $G_j \neq G_{j-1}$ there are at least $d \cdot e_j + \frac{1}{2}(d - e_j)(d + 1 - e_j)$ queries made less than estimated. For any of the possible values of e_j, this number is bounded from below by $\frac{1}{2}d(d+1)$. Furthermore, there are $\frac{1}{d+1}|G_n|$ such j and one has that the overall number of queries is at most

$$n \cdot d + |G_n| \cdot d - \tfrac{1}{d+1}|G_n| \cdot \tfrac{1}{2}d(d+1) = (n + \tfrac{1}{2}|G_n|) \cdot d.$$

It remains to show that the algorithm is correct, that is, that every isolated clique of size a or larger is covered by the conjectures C_i with $i \in D_n$.

So let C be an isolated clique with $|C| \geq a$. Note that whenever a set F_i intersects C then $F_i \subseteq C$: For every $j \in F_i$, $(i,j) \in E$ and thus $i \in C \Leftrightarrow j \in C$. Furthermore, one can show by induction that $|C \cap D_j| \leq 1$ for all j: This clearly holds for $j = 0$ and for all $j > 0$ where $|C \cap D_{j-1}| = 0$. So assume that $j > 0$ and $\{i\} = D_{j-1} \cap C$. Note that $D_j \subseteq D_{j-1} \cup \{j\}$, thus only the case $j \in C$ has to be considered. In this case $(i,j) \in E$ while $(i',j) \notin E$ for all $i' \in D_{j-1} - \{i\}$ by induction hypothesis. So the algorithm decides that $j \in F_i$ and $D_j = D_{j-1}$, in particular $|C \cap D_j| \leq 1$ again.

Whenever $G_j \supset G_{j-1}$ then $|D_{j-1} \cup \{j\}| = d + 1$ and $j \in F_j$. So either C

does not intersect D_{j-1} or $j \notin C$, thus $|C \cap (D_{j-1} \cup \{j\})| \leq 1$. It follows that only one of the $u_{i,j} \in G_j - G_{j-1}$ is in C. So, for all j, either $G_j = G_{j-1}$ or $|G_j - G_{j-1}| = d+1 \wedge |C \cap (G_j - G_{j-1})| \leq 1$. It follows that $|C \cap G_n| \cdot (d+1) \leq n$. Since d is the unique integer with $a \cdot d \leq n < a \cdot (d+1)$, $|C \cap G_n| < a$ and $C \not\subseteq G_n$.

Note that for all $j \in \{1, 2, \ldots, n\}$, $\{j\} \subseteq D_j - D_{j-1}$, $F_j \subseteq \{j, j+1, \ldots, n\}$ and $D_j \cap G_j = \emptyset$. If there is a j with $i \in D_{j-1} - D_j$ then $i \notin D_{k-1}$ for $k > j$ and thus $F_i \subseteq \{i, i+1, \ldots, j\}$. Furthermore, $i = u_{i,j} = \max\{ell \in F_i : \ell \leq j \wedge \ell \notin G_{j-1}\}$ and $F_i \subseteq G_j$. If $i \notin D_j$ for all j then $F_i = \emptyset$.

Since $C \not\subseteq G_n$, there is an F_i which intersects C. It follows that $F_i \not\subseteq G_n$, $i \in D_n$ and $F_i \subseteq C$. If $i' \in D_n$ then $F_{i'}$ is disjoint to C. So a $j \notin G_i$ is in C iff it is in F_i. Furthermore, all $j > i$ with $j \in C$ are also in F_i, so it is sufficient to query whether $(j, i) \in E$ for those j which are in G_n and which satisfy $j < i$. The resulting C_i is equal to C and the algorithm outputs a list containing all isolated cliques having at least a elements. ∎

Note that the upper bound on the number of queries done by this algorithm is almost optimal if one takes its behaviour on all graphs and not only on those graphs which have a sufficiently large clique. Taking a graph with n nodes and no edges such that $d \equiv n$ modulo $d+1$, the algorithm makes $\frac{d}{2}(n-1)$ queries in step (2) and $(n-d) \cdot d$ queries in step (3) resulting in $\frac{3}{2}n \cdot d - d^2 - \frac{d}{2}$ queries; so any reasonably improved upper bound on the number of queries has to exploit the fact that there is a clique of size a and might need some slight modifications of the algorithm.

Consider the case where d is given and a depends linearly on n through the equation $n = d \cdot a$. Then the upper bound on the queries is also linear in n and equals to $\frac{3 \cdot d}{2} \cdot n$. In particular, the classical case is covered by letting $d = 2$. For $d = 2$, the upper bound on the number of queries is $3 \cdot n$ which is roughly Blecher's bound doubled. But this bound also covers the case where the number of good and faulty processors can be equal. If at least a third of the processors is good and n is a multiple of 3, then the upper bound provided by the algorithm is $\frac{9}{2} \cdot n$.

As the relation $U(a, n) \leq \frac{3}{2 \cdot a} n^2$ shows that the upper bound is quadratic for fixed a, one might ask for the lower bound. The next theorem shows that lower and upper bound differ only by a factor 3 when n is sufficiently large compared to a. Furthermore, the bound is only slightly larger if more theories are supplied, thus it does make sense to restrict the search for the minimum necessary quantity of $\lfloor \frac{n}{a} \rfloor$ many theories.

Theorem 3.2. *Let $a \geq 2$. One needs at least $\frac{(n-a)(n-2a)}{2(a-1)} - b$ queries to find b sets such that every isolated clique is among these sets. In particular, $U(a, n) \geq \frac{1}{2(a-1)} n^2 - 3n$.*

Proof. Assume by way of contradiction that one would have an algorithm finding the desired b sets with $\lfloor \frac{(n-a)(n-2a)}{a-1} - b \rfloor$ many queries. Now one assumes that the algorithm is running for a graph with a single isolated clique C and that

G does not contain any further edge except the ones within C. The clique C is chosen adversary to the algorithm, so one assumes that the algorithm during the running time does not hit any edge of C and thus receives for every query the answer 0.

So let E' now be the set of all edges not queried. One has to show that E' contains at least $b+1$ cliques of size a and that thus the algorithm might fail to list the correct isolated clique C of G. Therefore, one looks at the b candidate cliques output by the algorithm and for each clique C' conjectured by the algorithm one removes one edge contained in C' from E' unless C' and E' do not have an edge in common. The resulting set E'' contains at least

$$\tfrac{n(n-1)}{2} - \tfrac{(n-a)(n-2a)}{a-1}$$

edges. This an upper bound on the number $\frac{n(n-1)}{2} - t_{a-1}(n)+1$ given by Bollobás where Bollobás [5, VI.1,Theorem 1.1] showed that such graphs having so many edges and n nodes have at least one clique of size a or more. So one can choose a clique C of the graph $(\{1,2,\ldots,n\},E'')$. This clique C has at least a nodes and the corresponding graph G consisting of the isolated clique C is then the adversary graph on which the given algorithm fails. ∎

4 Explicit Bounds for U(a,n) with a = 2, 3, ..., 9

In the following one might ask what bounds one can obtain for fixed values for a where n remains unspecified. It is shown that the upper bound $\frac{2}{a}n^2$ is not optimal for certain a and explicit better bounds are given. In the case of $a=2$ the exact number of queries needed to find the isolated cliques of size 2 can be determined.

Theorem 4.1. $U(2,n) = \lceil \frac{n(n-2)}{2} \rceil$.

Proof. For $U(2,n) \geq \lceil \frac{n(n-2)}{2} \rceil$, simulate any correct algorithm M by answering "$(i,j) \notin E$" to every query asked until M stops. Then for every query (i,j) where it had not been asked whether $(i,j) \in E$, one considers the corresponding isolated clique $\{i,j\}$ of the given graph. Each graph containing exactly the edge (i,j) and no other one would be consistent with the data supplied to M. As M outputs at most $\lfloor \frac{n}{2} \rfloor$ many hypotheses, one knows that at least $\lceil \frac{n(n-2)}{2} \rceil$ queries have been asked.

For the converse direction, let $(\{1,2,\ldots,n\},E)$ be the given graph. Now one asks whether $(i,j) \in E$ for all i,j where either i is even and $j \geq i+1$ or i is odd and $j \geq i+2$. These are $\lceil \frac{n(n-2)}{2} \rceil$ many queries. Now one shows that there are at most $\lfloor \frac{n}{2} \rfloor$ many isolated cliques of size 2 or more which are consistent with the answers so one can output these cliques without violating the bound on the number of hypotheses.

So assume that C, C' are two cliques of size 2 or more which are consistent with all answers supplied. Furthermore, assume that C, C' have a common node and that there is a node which is in one but not in both cliques. So let i, j be these nodes with $i < j$, $\{i, j\} \subseteq C$ and $\{i, j\}$ intersecting but not being a subset of C'. The query whether $(i, j) \in E$ cannot have been asked as it is true for C and false for C', thus $j = i + 1$ and i is odd. As the cardinality of C' is at least 2, there is some node $k \in C' - \{i, j\}$. Both queries whether $(i, k), (j, k) \in E$ (in case $i, j < k$) or whether $(k, i), (k, j) \in E$ (in case $k < i, j$) have been asked. As C contains i, j, both answers have to be the same. As C' contains k and one of the elements i, j, one of the answers must be true and the other one false. It follows that, whatever answers had been supplied, only one of the candidates C, C' can be consistent with these answers.

So one can derive that whenever C, C' are both candidates for isolated cliques of size 2 and more which are consistent with all answers to queries of the algorithm, then C and C' have to be disjoint. So there can be at most $\lfloor \frac{n}{2} \rfloor$ candidates consistent with the answers to all queries and $U(2, n) \leq \lceil \frac{n(n-2)}{2} \rceil$. ∎

Although one cannot verify with few queries, whether a given candidate for an isolated clique is really one, one might at least try to get more evidence. This might need more queries and so the following notion $U^*(a, n)$ captures this concept. The motivation for this notion is, that it is easier to iterate procedures searching the isolated cliques if the additional constraints (B) and (C) are met.

Definition 4.2. The notion $U^*(a, n)$ is the amount of queries necessary to come up with the set of all candidates $C_1, C_2, \ldots, C_e$ the three conditions below hold where E' be the set of edges for which the algorithm asked whether they are in E or not.

(A) Every set C_i has at least a members and $e \leq \frac{n}{a}$.
(B) For every $i \in \{1, 2, \ldots, e\}$, the subgraph containing the nodes of C_i and those edges of E' which connect two nodes in C_i is connected.
(C) For every distinct $i, j \in \{1, 2, \ldots, e\}$, there are nodes $i' \in C_i$ and $j' \in C_j$ such that the edge from i' to j' is in E'.

Note that the conditions (B) and (C) enforce that all candidates for cliques produced by the algorithm are disjoint.

As obviously $U(a, n) \leq U^*(a, n)$, the below upper bounds are stated for $U^*(a, n)$ and apply to $U(a, n)$, too.

Theorem 4.3. $U^*(a + b - 1, n + m) \leq U^*(a, n) + U^*(b, m) + \frac{m \cdot n}{\max\{a,b\}}$.

Proof. Given a graph $(\{1, 2, \ldots, n + m\}, E)$, one first uses the subroutines needing $U^*(a, n) + U^*(b, m)$ many queries in order to find candidates $A_1, \ldots, A_c$ and $B_1, \ldots, B_d$ for isolated cliques of size a and b, respectively, in the subgraphs containing the edges of E within the sets $\{1, 2, \ldots, n\}$ and $\{n+1, n+2, \ldots, n+m\}$, respectively. Note that $c \leq \frac{n}{a}$ and $d \leq \frac{m}{b}$. Let $x_1, x_2, \ldots, x_c$ and $y_1, y_2, \ldots, y_e$ be members of these candidates and let A_0 and B_0 be the sets of those elements of $\{1, 2, \ldots, n\}$ and $\{n + 1, n + 2, \ldots, n + m\}$ which are not contained in any of the candidates for cliques. Now one asks the following queries:

– Is $(x_i, y_j) \in E$ for $i \in \{1, 2, \ldots, c\}$ and $j \in \{1, 2, \ldots, d\}$;
– Is $(x_i, y) \in E$ for $i \in \{1, 2, \ldots, c\}$ and $y \in B_0$;
– Is $(x, y_j) \in E$ for $x \in A_0$ and $j \in \{1, 2, \ldots, d\}$.

Assume now that $a \leq b$. The first two groups of queries have at most cm queries, the third group $|A_0|\frac{m}{b}$ queries as $d \leq \frac{m}{b}$. Using that $c \leq \frac{1}{a}(n - |A_0|)$ and $\frac{1}{b} \leq \frac{1}{a}$ one obtains that the number of queries has the upper bound $\frac{1}{a}(n - |A_0|)m + \frac{1}{b}|A_0|m \leq \frac{mn}{a}$. In the case that $a > b$ one can similarly show that the number of queries is bounded by $\frac{mn}{b}$. Together with the $U^*(a, n) + U(m, n)b$ many queries from the built in subroutines, this gives the bound on the number of queries in the statement of the theorem.

If C is an isolated clique containing at least $a + b - 1$ nodes, than either $C \cap \{1, 2, \ldots, n\}$ has at least a or $C \cap \{n + 1, n + 2, \ldots, n + m\}$ has at least b nodes. So at least one of these sets is an isolated clique in the subgraph having at least a or b nodes, respectively. It follows that C restricted to one of the subgraphs is in the candidates $A_1, A_2, \ldots, A_c$ or $B_1, B_2, \ldots, B_d$ and so shows up in the list of candidates. Condition (A) is satisfied.

It remains to show conditions (B) and (C). Say that $C \cap \{1, 2, \ldots, n\} = A_1$. Then the algorithm establishes whether one of the B_k with $k = 1, 2, \ldots, d$ belongs to C. If so, $C = A_1 \cup B_k$ and the set of edges queried by the algorithm is connected when restricted to A_1 or B_k. If not, $C \subseteq A_1 \cap B_0$ and as all queries (x_1, y) with $y \in B_0$ had been made, again the queries asked by the algorithm inside C form a connected subgraph and (B) holds.

To verify (C), fix now that C extends A_1. By induction hypothesis, $C \cap \{1, 2, \ldots, n\} = A_1$ and it holds that for each clique C' containing one of the sets $A_2, \ldots, A_c$ there is an edge having a node in A_1 and one in C' which had been queried by the algorithm. So consider any further clique C' containing some set B_k. As the query (x_1, y) had been asked explicitly, again (C) is satisfied for C'. ∎

Corollary 4.4. *$U^*(2a, n) \leq U^*(a, \gamma n) + U^*(a+1, (1-\gamma)n) + \frac{\gamma(1-\gamma)}{a}n^2$ whenever γn is an integer and $U^*(2a - 1, n) \leq 2 \cdot U^*(a, \frac{1}{2} \cdot n) + \frac{1}{4a}n^2$ whenever n is even.*

Proposition 4.5. *$U^*(bc + 1, n) \leq cU^*(b + 1, \frac{1}{c}n) + \frac{c-1}{2(b+1)c}n^2$ whenever n is a multiple of c.*

Proof. The proof is parallel to that of Theorem 4.3. One divides the graphs into c groups of $\frac{n}{c}$ nodes and establishes on them all candidates for isolated cliques of size $b + 1$. Having those, one has now to combine every candidate of a clique with nodes from the other groups which gives the bound of $\frac{(c-1)c}{2} \cdot \frac{1}{b+1} \cdot (\frac{n}{c})^2$ many further queries. By multiplying these terms, one has the number of queries in the Proposition. ∎

Now one can use these bounds in order to compute explicit bounds for the constant $a = 3, 4, 5, \ldots, 9$. Note that $U^*(2, n) = \frac{n(n-1)}{2}$, which is almost but not exactly the value of $U(2, n)$. The lower bounds come from Theorem 3.2, the upper bound is obtained by using Corollary 4.4 iteratively. In order to make computations a bit easier and to ignore the effects of iterated rounding, the below table gives the values for those n which are multiples of $2^3 \cdot 3^2 \cdot 17$. These n can be divided whenever a problem is split into subproblems as for example the algorithm for $U^*(6,n)$ solves the subproblems belonging to $U^*(3, \frac{4}{9}n)$ and $U^*(4, \frac{5}{9}n)$ first and recombines them later.

This and similar results can be applied to get the following explicit values for small constants a. Note that the condition that n is a multiple of 1224 only applies in order to reduce the number of case distinctions in notation and proofs.

Theorem 4.6. *Let n be a multiple of 1224. Then*

$$\begin{array}{llll}
0.2500n^2 - 3n \le U^*(3,n) \le 2 \cdot U^*(2, \frac{1}{2}n) + \frac{1}{8}n^2 & \le \frac{3}{8}n^2 & \le 0.3750n^2, \\
0.1666n^2 - 3n \le U^*(4,n) \le U^*(2, \frac{1}{3}n) + U^*(3, \frac{2}{3}n) + \frac{1}{9}n^2 & \le \frac{1}{3}n^2 & \le 0.3334n^2, \\
0.1250n^2 - 3n \le U^*(5,n) \le 2 \cdot U^*(3, \frac{1}{2}n) + \frac{1}{12}n^2 & \le \frac{13}{48}n^2 & \le 0.2709n^2, \\
0.1000n^2 - 3n \le U^*(6,n) \le U^*(3, \frac{4}{9}n) + U^*(4, \frac{5}{9}n) + \frac{20}{243}n^2 & \le \frac{7}{27}n^2 & \le 0.2593n^2, \\
0.0833n^2 - 3n \le U^*(7,n) \le 2 \cdot U^*(4, \frac{1}{2}n) + \frac{1}{16}n^2 & \le \frac{11}{48}n^2 & \le 0.2292n^2, \\
0.0714n^2 - 3n \le U^*(8,n) \le U^*(4, \frac{7}{17}n) + U^*(5, \frac{10}{17}n) + \frac{35}{578}n^2 & \le \frac{731}{3468}n^2 & \le 0.2108n^2, \\
0.0625n^2 - 3n \le U^*(9,n) \le 2 \cdot U^*(5, \frac{1}{2}n^2) + \frac{1}{20}n^2 & \le \frac{89}{480}n^2 & \le 0.1855n^2.
\end{array}$$

In particular $r_a n^2 - 3n \le U(a,n) \le q_a n^2$ where $r_a n^2 - 3n$ and $q_a n^2$ are the entries in the first and last column, respectively, for $a = 3, 4, \ldots, 9$.

Note that Proposition 4.5 gives for $a = 4$ by the relation $U^*(4,n) \le 3 \cdot U^*(2, \frac{1}{3}n) + \frac{1}{6}n^2 \le \frac{1}{3}n^2$ the same upper bound as the application of Corollary 4.4, but for $a = 5, 6, 7, 8, 9$, the method above turned out to be superior than the one given by Proposition 4.5. On the other hand, Theorem 3.1 gives better bounds for $a \ge 6$, so that this general algorithm beats the solutions optimized for these specific values of a; the corresponding upper bound $\frac{3}{2a} \cdot n^2$ is approximately $02500 \cdot n^2$, $0.2143 \cdot n^2$, $0.1875 \cdot n^2$, $0.1667 \cdot n^2$ for $a = 6, 7, 8, 9$, respectively. The main value of the techniques in the present section is that they other than Theorem 3.1 can be parallelized as outlined in the next section.

5 Parallel Queries

For the case that $a > \frac{n}{2}$, it has been shown that one can parallelize the search for a isolated clique (there is at most one) in the sense that in every round one is permitted to ask as many questions as long no two of these questions deal with edges having a node in common. It had been shown that one can then find the isolated clique with a constant number of rounds [2, 3, 8]. In the following, it is shown to which extent the previous results can be transferred to the model of parallel queries.

Theorem 5.1. *If $(\{1, 2, \ldots, n\}, E)$ is a directed graph, then the number of rounds to find all isolated cliques of size a or more is at most $2 \cdot PU(a, n)$.*

Proof. Let $F = \{(i,j) : (i,j) \in E \wedge (j,i) \in E\}$. Note that if C is an isolated clique with respect to E then C is also an isolated clique with respect to F. As one can simulate all queries to E by two queries to F involving the same nodes, one can simulate an m-round algorithm for F by an $2m$-round algorithm for the graph with edges E and afterwards copycat the output. ∎

Therefore it is sufficient to consider the case of undirected graphs only. The lower bound for $U(a,n)$ from Theorem 3.2 translates into a lower bound for $PU(a,n)$ as every query involves two nodes and thus at most $\frac{n}{2}$ many queries can be done in parallel.

Corollary 5.2. *$PU(a,n) \geq \frac{1}{a-1}n - 6$.*

The authors did not find a way to turn the algorithm from Theorem 3.1 into a parallel one. Nevertheless, there is a parallel and iterated version of the approach from Theorem 4.3 giving the following result.

Theorem 5.3. *$PU(a,n) \leq 6\frac{\log(a)+2}{a}n$.*

The original setting was for directed graphs, but as one can ask both directions in two levels after each other, one obtains in general, that the number of rounds for the directed graphs is at most the double of the number of rounds for the undirected graphs. So, in the following, only the numbers of rounds for undirected graphs are analyzed.

The constant number of rounds was of course only possible for the case that $a \geq \gamma n$ for some constant γ as Theorem 3.2 states, that at least $\frac{1}{2(a-1)}n^2 - 3n$ queries are necessary, that implies, at least $\frac{n}{a-1} - 6$ rounds. The algorithm given in Theorem 3.1 is extremely non-parallel. Therefore, it is more convenient to try to adapt the results from Section 4 in order to get upper bounds on the number of rounds required.

Definition 5.4. Let $PU(a,n)$ be the number of rounds necessary to find all isolated cliques of size a or more in a graph of n nodes. Let $PU^*(a,n)$ be the corresponding number if in addition the conditions (A), (B) and (C) from Definition 4.2.

Proposition 5.5. *$PU^*(2a+1,n) \leq PU^*(a+1,\frac{1}{2}n) + 2 + \frac{2n}{a+1}$ if n is even.*

Proof. The algorithm is the same as in Theorem 4.3, so the main question is how to make the queries parallel. As the two subproblems $PU^*(a+1,\frac{1}{2}n)$ work on disjoint sets of nodes, they do not interfere with each other. Combining the results, one should note that every query involves on at least one side a connected candidate clique containing at least $a+1$ nodes. As it does not matter which node is taken, one can parallelize the queries in the three steps of the combination as follows.

Recall that E is the set of edges, $A_1,\ldots,A_c$ are the candidate cliques from one side and $B_1,\ldots,B_d$ those from the other sides. The nodes in A_0 and B_0 do not belong to any candidate clique, all candidate cliques are disjoint and have at least size $a+1$. The queries are the following.

– Is $(x_i, y_j) \in E$ for $i \in \{1, 2, \ldots, c\}$, $j \in \{1, 2, \ldots, d\}$;
– Is $(x_i, y) \in E$ for $i \in \{1, 2, \ldots, c\}$ and $y \in B_0$;
– Is $(x, y_j) \in E$ for $x \in A_0$ and $j \in \{1, 2, \ldots, d\}$.

In Theorem 4.3 the x_i are the minimum of A_i and y_j the minimum of B_j. When parallelizing the queries, this is a bad choice. Indeed it does not matter which $x_i \in A_i$ and $y_j \in B_j$ are taken, son one distributes the taken queries such that every node receives more or less the average load. Therefore the queries of the first group can be done in $1 + \frac{max\{c,d\}}{a+1}$ many rounds. The queries of the second group need at most $1 + \max\{c, \frac{n}{a+1}\}$ many rounds as one can access in each round for each A_i in parallel $a+1$ members of B_0 as long as these members are not overcrowded by several $A_i, A_{i'}$ trying to access them. The third group needs at most $1 + \max\{d, \frac{n}{a+1}\}$ many rounds and can be carried out in parallel to the second group, as the second group deals with edges between nodes in some A_i, $i \geq 1$ and B_0 while the third group with edges between nodes A_0 and some B_j, $j \geq 1$. As $c(a+1) \leq n$ and $d(a+1) \leq n$, one gets the upper bound $2 + \frac{2n}{a+1}$. ∎

Theorem 5.6. *$PU(a, n) \leq 4\frac{\log(a)+2}{a}n$ if n is a multiple of $2^{\log(a)+1}$.*

Proof. Assume that $2^b + 1 \leq a \leq 2^{b+1}$ and therefore $b \leq \log(a) + 1$. The smaller a is, the more rounds are needed, so one can without loss of generality consider the case that $a = 2^b + 1$. Note that Proposition 5.5 implies that

$$PU^*(2^{b-k} + 1, 2^{-k}n) \leq PU^*(2^{b-k-1} + 1, 2^{-k-1}n) + 2^{1-k}n/(2^{b-k} + 1)$$

for $k = 0, 1, 2, \ldots, b-1$ and using that $2^{1-k}/(2^{b-k}+1)$ is below 2^{1-b} one obtains that

$$PU^*(2^b + 1, n) \leq PU^*(2, 2^{-b}n) + b \cdot n \cdot 2^{1-b}.$$

As $PU^*(2, 2^{-b}n)$ is 2^{1-b} one obtains the relation

$$PU(a, n) \leq 2^{1-b} \cdot (1 + b) \cdot n$$

and by $2^{1-b} \leq \frac{4}{a}$ and $1 + b \leq \log(a) + 2$ one obtains the bounds in the statement of the Theorem. ∎

So $PU(a, n) \in O(\frac{\log(a)}{a}n)$. It is an open problem whether one can find the isolated cliques even in $O(\frac{1}{a}n)$ rounds.

Acknowledgments. The authors would like to thank Richard Beigel and Wolfgang Merkle for comments.

References

1. Ferrucio Barsi, Fabrizio Grandoni and Piero Maestrini. A theory of diagnosability of digital systems. *IEEE Transactions on Computers*, 25:585–593, 1976
2. Richard Beigel, S. Rao Korsaraju and Gregory F. Sullivan. Locating Faults in a Constant Number of Parallel Testing Rounds. *Proceedings of the ACM Symposium on Parallel Algorithms and Architectures*, pages 189–198, 1989.

3. Richard Beigel, William Hurwood and Nabil Kahale. Fault diagnosis in a flash. *Proceedings f the 36th IEEE Symposium on Foundations of Computer Science*, pages 571–580, 1995.
4. Pavel M. Blecher. On a Logical problem. *Discrete Mathematics*, 43, 107–110, 1983.
5. Béla Bollobás. *Extremal Graph Theory.* Academic Press, London, 1978.
6. Béla Bollobás. *Combinatorics.* Cambridge University Press, Cambridge, 1986.
7. Kyung-Yong Chwa and S. Louis Hakimi. Schemes for fault-tolerant computing: A comparison of modularly redundant and t-diagnosable systems. *Information and Control*, 49:212–238, 1981.
8. William Hurwood. *System level fault diagnosis under static, dynamic, and distributed models.* PhD Dissertation, Yale, New Haven, CT, 1996.
9. Miroslaw Malek. A comparison connection assignment for diagnosis of multiprocessor systems. *Proceedings of the 7th Annual Symposium on Computer Architecture*, La Baule, USA, pages 31–36, 1980.
10. Andrzej Pelc. Undirected graph models for system-level fault diagnosis. *IEEE Transactions on Computers*, 40:12711276, 1991.
11. Andrzej Pelc. Optimal fault diagnosis in comparison models. *IEEE Transactions on Computers*, 41:779–786, 1992.
12. Franco P. Preparata, Gernot Metze and Robert T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Transactions on Electronic Computers*, 16:848-854, 1967.